流水线设计思路

统一 Clock

Step1：Fetch Instructions

器件操作

Mux：选择指令
Add：PC + 4 并写回 PC 为下一个时钟周期做好准备
指令寄存器：根据 PC 送来的地址进行取址

Reg1：IF / ID – Store

1. 保存取址得到的指令
2. 32 位 PC 自增地址（Add 完的 PC+4）

Step2：Decode

Reg1：IF / ID – Read

1. 将后 16 位立即数送入符号扩展单元
2. 将 rs（旁路用）、rt、rd 寄存器号送入寄存器堆

器件操作

每条指令在译码阶段都要判断
根据 IF/ID 寄存器传来的值进行读寄存器中的数据
对指令后 16 位进行符号扩展

冒险检测单元：插入阻塞
条件：P212

Reg2：ID / EXE – Store

1. 寄存器读来的数据 1
2. 寄存器读来的数据 2
3. 符号扩展单元传来的扩展完的 32 位数据
4. PC 自增值
5. 写寄存器传入 ID/EX

Step3：Execution

Reg2：ID / EXE – Read

1. 寄存器 1 传来的数据送入 ALU_Source_A
2. 寄存器 2 传来的数据送入 ALU_Source_B_Mux
3. 寄存器 2 传来的数据送入 Reg_EX/MEM

4. 符号扩展单元传来的数据送入左移单元
5. 符号扩展单元传来的数据送入 ALU_Source_B_Mux

器件操作

ALU：对 Source_A 和 Source_B 传来的数据进行相加

Reg3：EXE / MEM

1. 将 ALU_Out_Zero 存入 Reg_EX/MEM
2. 将 ALU_Out_C 存入 Reg_EX/MEM
3. 写寄存器传入 EX/MEM（from ID）

Step4：Memory Access

Reg3：EXE / MEM

1. 从 Reg_EX/MEM 中得到的地址读取数据存储器（from ALU_Out_C）
2. 从 Reg_EX/MEM 中得到的数据写入地址对应的数据存储器（from RF_Out_2）

器件操作

数据存储器：根据传入的地址及数据进行写入 or 读取

Reg4：MEM / WB

1. 将数据存储器中读出的数据写入 Reg_MEM/WB
2. 将 ALU_Out_C 的数据传入 Reg_MEM/WB
3. 写寄存器传入 EX/MEM（from ID）

Step5：Write Back

Reg4：MEM / WB

1. 将从数据存储器中读出的数据传入 RegWriter_Mux
2. 将从 ALU_Out_C 的数据传入 RegWriter_Mux
3. 将写寄存器传入寄存器堆的写寄存器端口（from ID）

器件操作

寄存器堆：根据写寄存器地址及数据进行 Write Back

Hazard

1. 结构冒险 Structural Hazard

Occasion

硬件资源竞争，IF 阶段和 MEM 阶段同时进行内存访问

Solvation

指令、数据内存分开

2. 数据冒险 Data Hazard

Occasion

上一个周期写回 WB 和这一个周期 EX 共用一个寄存器

Solvation

Forwarding，数据直接传递给下一个指令

A. 运算依赖，R-Type → R-Type：EX_1_Out to EX_2_In

B. 访存依赖，lw → R-Type：MEM_1_Out to EX_2_In，仍然需要 nop 来 bubble 一个周期

旁路 & 阻塞

(1) 旁路单元：判断冒险条件

a. 当前面指令有写寄存器时才发生 forwarding

b. Rd≠$Zero

c. 判断条件

i. （A）EX/MEM.Register_**Rd** = ID/EX.Register_**Rs** or ID/EX.Register_**Rt**
Forwarding from **EX/MEM.Register** to **ALUSource.Mux**

ii. （B）MEM/WB.Register_**Rd** = ID/EX.Register_**Rs** or ID/EX.Register_**Rt**
Forwarding from **MEM/WB.Register** to **ALUSource.Mux**

(2) 冒险检测单元：

a. ID/EX.MemRead

b. ID/EX.Register_Rt = IF/ID.Register_Rs

c. ID/EX.Register_Rt = IF/ID.Register_Rt

d. Output to：若满足则插入 bubble，指令被延后阻塞一周期后，旁路单元可正常工作；处于 ID 级指令被阻塞，IF 也被阻塞；保持 PC、IF/ID.Register 不变（插入 nop）

i. PCWrite（拒绝更新）

ii. IF/IDWrite

iii. ID/EX.Register.Mux（清空）

实现：

a. 强制 ID/EX.Register = 0，则 EX、MEM、WB 为 nop

b. 防止 PC、IF/ID.Register 更新（保持现状）→ 一周期阻塞后可继续进行（lw 要到 EX）

3. 控制冒险 Control Hazard

Occasion

决策依赖于一条指令的结果（beq/bne 跳转地址取址地址在 EXE 后才知道）

Solvation_1

Branch → Any：ID_Special_1_Out to IF_2（提到 ID 阶段计算跳转地址 & ID 阶段减法直接判断，仍然需要 nop 来 bubble 一个周期

Solvation_2

预测 / 有限状态自动机

流水线寄存器

控制信号携带

IF/ID → Control 单元解析 → ID/EX （WB/MEM/EX）→ EX/MEM （WB/MEM）→ MEM/WB （WB）

旁路与阻塞

控制信号 P204 写入有效/无效时区=-0

Step1_IF

读指令存储器和写 PC 的控制信号总是有效的

Step2_ID

Step3_EX

RegDst 选择结果寄存器
ALUOp ALU 操作
ALUSrc 为 ALU 读取数据 2 或符号扩展后的立即数

Step4_MEM:

Branch 相等则分支
MemRead 装载指令
MemWrite 存储指令
除非控制电路确定是一条分支指令并且 ALU 结果为 0，否则将选择线性地址中的下一条指令作为 PCSrc 信号

Step5_WB

MemtoReg 决定将 ALU 结果 or 存储器数据传送到寄存器堆
RegWrite 决定是否写入寄存器堆